last-lab

Хийсвэр функц гэж юу вэ? Тодорхойлолт, зарлалт, жишээ, давуу талыг тус тус бич.

C++ хэл дээр хийсвэр функц нь зарласан боловч тодорхойлогдоогүй функц юм. Хийсвэр функцийг ихэвчлэн хийсвэр классын нэг хэсэг болгон ашигладаг бөгөөд энэ нь давхардлаас сэргийлэх давуу талтай. Эх класс нь interface маягтай бөгөөд үүнээс удамших классуудад энэ функцыг тодорхойлж болно. Мөн бичдэс нь өөр адил нэртэй функц тодорхойлох боломжийг олгодог.

Жинхэнэ хийсвэр функц гэж юу вэ? Тодорхойлолт, зарлалт, жишээ, давуу талыг тус тус бич.

C++ хэл дээр хийсвэр функц нь зарласан боловч тодорхойлогдоогүй функц юм. Хийсвэр функцийг ихэвчлэн хийсвэр классын нэг хэсэг болгон ашигладаг бөгөөд энэ нь давхардлаас сэргийлэх давуу талтай. Эх класс нь interface маягтай бөгөөд үүнээс удамших классуудад энэ функцыг тодорхойлж болно. Мөн бичдэс нь өөр адил нэртэй функц тодорхойлох боломжийг олгодог.

PURE VIRTUAL нь function() = 0; байна.

```
class Shape {
public:
    virtual void draw() = 0; // pure virtual function
};

class Circle : public Shape {
public:
    void draw() override {
        // implementation for drawing a circle
    }
};

class Square : public Shape {
public:
    void draw() override {
        // implementation for drawing a square
    }
};
```

Функц дахин программчлах гэж юу вэ? Эх классын дахин программчилсан функцыг хүүхэд классын функц дотроос хэрхэн дууддаг вэ?

Хийсвэр класс ашиглан үүсгэсэн классууд дотор эх классын гишүүн функцыг дахин программчилдаг ба, энэ нь эх классынхаа бичдэстэй яг адилаар бичигдсан байх ёстой.

virtual түлхүүр үг ашиглан үүсгэсэн функцыг хүүхэд класс дээр нь override гэж ахин тодорхойолох боломжтой ба хэрвээ эх классынхаа функцыг дуудах хэрэг гарвал

`BaseClass::FunctionName();` гэх хэлбэрээр дуудна.

```
class Shape
{
public:
    virtual void draw()
    {
        std::cout << "Drawing a shape" << std::endl;
    }
};

class Circle : public Shape
{
public:
    void draw() override
    {
        std::cout << "Drawing a circle" << std::endl;
    }
};
```

Хийсвэр класс гэж юу вэ? Хэрхэн объект байгуулдаг вэ?

Хийсвэр класс гэдэг нь тухайн классыг шууд ашиглан обьект үүсгэх боломжгүй, зөвхөн удамшуулан хэрэглэх зориулалттай класс юм.

Энэ класс дотор virtual function тодорхойлж удамших классууд дээр хэрэгжүүлдэг.

Удамшилд байгуулагч функц хэрхэн ашиглагддаг вэ? Удамшил ба байгуулагч функцийн хүрээнд үзсэн зүйлүүдийг бич.

Удамшиж үүссэн классаас өөрийн гэсэн байгуулагч функцтай байх ба хэрвээ эх классынхаа байгуулагч функцыг дуудах хэрэг гарвал `:` тэмдгийг өөрийн байгуулагчийн ардаас дуудаж ашиглана.

```
class Parent
{
public:
    Parent()
    {
        // Initialize parent class's properties
    }
};

class Child : public Parent
{
public:
    Child() : Parent()
    {
        // Initialize child class's properties
    }
};
```

Удамшил ба устгагч функц хоёр ямар хамааралтай вэ?

Ямарваа нэг класс өөр нэг классаас удамшихад тухайн класст өөр нэгэн устгагч функц үүсэх ба энэ устгагч функц удамшуулсан эх классынхаа устгагч функцыг дууддаг байна.

Бодлого:

Lab07-д хийсэн дүрсүүдэд жинхэнэ хийсвэр функц нэмж
a. Shape класст примтетер олох функцыг жинхэнэ хийсвэрээр зарлана,
b. TwoDimensionalShape класст талбай олох функцийг жинхэнэ хийсвэрээр зарлана.

```
#include <iostream>
using namespace std;

class shape {

public:
    //virtual function calculates perimeter
    virtual double getPerimeter() = 0;
    virtual string getName() = 0;

};
```

```
class twoDimensionalShape : public shape {

public:
    // virtual function calculates area
    virtual double getArea() = 0;

};

int main() {


  return 0;
}
```

Хийсвэр эх классын параметертэй, параметергүй байгуулагч функцийг хүүхэд класс бүрийн байгуулагч функцтай давхар дууддаг болгоно.

```
#include <iostream>
using namespace std;

class shape {

public:
    //virtual function calculates perimeter
    virtual double getPerimeter() = 0;
    shape(int a){
        cout << "Shape constructor called" << endl;
    }

};

class twoDimensionalShape : public shape {

public:
    // virtual function calculates area
    virtual double getArea() = 0;

};

class Triangle : public shape {
private:
    double base;
    double height;
public:
// Эх классыг дуудаж байна
    Triangle(double b, double h) : shape(1) {
        base = b;
        height = h;
```

```
    }
    double getPerimeter() {
        return base + height + sqrt(base*base + height*height);
    }
};

int main() {


  return 0;
}
```

Харилцан адилгүй талбайтай олон ширхэг тойрог, гурвалжин, тэгш өнцөгтүүд үүсгэж хооронд нь талбайгаар нь эрэмбэл.

```
#include<iostream>
#include <string.h>
#include <math.h>
using namespace std;
class twoD{
    private:
    int number ;
    protected:
        int x1 , y1 ;
        int length ;
        float s;
        char *name;
    public:
         twoD();
        void set_number();
        void show_number();
        void set_name( char *name);
        char *get_name();
        ~twoD();
        virtual float area();
        virtual float theLength();
        virtual void point();
};
class circle: public twoD{
    public:
        circle(int x1 , int y1 , int length);
        float area();
        float theLength();
};
class kvadrat: public twoD{
    private:
        int x2 ,x3 , x4 , y2 , y3 , y4;
    public:
        kvadrat(int x1 , int y1 , int length);
        float area();
```

```
        float theLength();
        void point();
};
class triangle: public twoD{
    private:
        int x2 ,x3 , y2 , y3 ;
    public:
        triangle(int x1 , int y1 , int length);
        float area();
        float theLength();
        void point();
};

#include "header.h"
        circle::circle(int x1 , int y1 , int length){
            this->x1 = x1;
            this->y1 = y1;
            this->length = length;
        }
        float circle::area(){
            return this->s =  3.14*length*length;
        }
        float circle::theLength(){
          return  2* 3.14*length ;
        }
        kvadrat::kvadrat(int x1 , int y1 , int length){
            this->x1 = x1;
            this->y1 = y1;
            this->length = length;
        }
        float kvadrat::area(){
          return this->s =  length*length;
        }
        float kvadrat::theLength() {
              return 3*length ;
        };
        void kvadrat::point(){
            this->y2 = y1;
            this->x4 = x1;
            this->x2 = this->length + x1;
            this->y4 = this->length + y1;
            this->x3 = this->length + this->x4 ;
            this->y3 = this->length + this->y2;
            cout << "1-r oroi "<< x1 << " " << y1 << endl ;
            cout << "2-r oroi " << x2 << " "  << y2 << endl ;
            cout << "3-r oroi " << x3  << " " << y3 << endl ;
            cout << "4-r oroi " << x4  << " " << y4 << endl ;
        }
        triangle::triangle(int x1 , int y1 , int length){
            this->x1 = x1;
            this->y1 = y1;
            this->length = length;
        }
        float triangle::area(){
```

```
          return  sqrt(3)*length*length / 4;
        };
        float triangle::theLength() {
              return 3*length ;
        };
        void triangle ::point(){
            this->y2 = this->y1 + length* sqrt(3)/2;
            this->x2 = this->length/2 + x1;
            this->x3 = this->length/2 - x1;
            this->y3 = this->y1 + length* sqrt(3)/2;
            cout << "1-r oroi "<< x1 << " " << y1 << endl ;
            cout << "2-r oroi " << x2 << " "  << y2 << endl ;
            cout << "3-r oroi " << x3  << " " << y3 << endl ;
        }

        twoD::twoD(){
            number++;
         }
         twoD::~twoD(){
            number--;
        }
        void twoD::set_number(){
            number = 0 ;
        }
        void twoD::show_number(){
            cout <<  number;
        }
        void twoD::set_name( char *name){
            delete name ;
            name = new char[strlen(name) + 1];
            strcpy(this->name , name);
        }
        char *twoD::get_name(){
return name;
        }
        float twoD::area(){
            return this->s;
        }

int main(){
    circle c1(0,0,1), c2(0,0,1);
    triangle t1(0,2,3), t2(1,2,2);
    kvadrat k1(1,2,3), k2(3,4,2);
    twoD *shape[6];
    shape[0] = &c1;
    shape[1] = &c2;
    shape[2] = &t2;
    shape[3] = &t2;
    shape[4] = &k2;
    shape[5] = &k2;
    for(int j=1;j<6;j++){
        twoD *a;
        int x=shape[j]->area();
        int i=j-1;
```

```
         while(i>=0 && shape[i]->area() > x){
             a = shape[i];
             shape[i+1]=shape[i];
             i = i-1;
         }
         shape[i+1] = a;
    }
    for(int i=0 ; i<6; i++)
cout<<shape[i]->area()<<endl;
    for(int j=1; j<6; j++){
       twoD *a;
       int x=shape[j]->theLength();
       int i=j-1;
       while(i>=0 && shape[i]->theLength() > x){
             a = shape[i];
             shape[i+1]=shape[i];
             i=i-1;
         }
         shape[i+1] = a;
    }
     for(int i=0 ; i<6 ; i++) cout<<shape[i]->theLength()<<endl;
     shape[6]->set_number();
     shape[6]->show_number();
    return 0;
}
```